

DIVIDE-AND-CONQUER COMPUTATION OF CYLINDRICAL ALGEBRAIC DECOMPOSITION

ADAM STRZEBOŃSKI

ABSTRACT. We present a divide-and-conquer version of the Cylindrical Algebraic Decomposition (CAD) algorithm. The algorithm represents the input as a Boolean combination of subformulas, computes cylindrical algebraic decompositions of solution sets of the subformulas, and combines the results using the algorithm first introduced in [34]. We propose a graph-based heuristic to find a suitable partitioning of the input and present empirical comparison with direct CAD computation.

1. INTRODUCTION

A *real polynomial system* in variables x_1, \dots, x_n is a formula

$$S(x_1, \dots, x_n) = \bigvee_{1 \leq i \leq l} \bigwedge_{1 \leq j \leq m} f_{i,j}(x_1, \dots, x_n) \rho_{i,j} 0$$

where $f_{i,j} \in \mathbb{R}[x_1, \dots, x_n]$, and each $\rho_{i,j}$ is one of $<, \leq, \geq, >, =$, or \neq .

A subset of \mathbb{R}^n is *semialgebraic* if it is a solution set of a real polynomial system.

Every semialgebraic set can be represented as a finite union of disjoint *cells* (see [21]), defined recursively as follows.

- A cell in \mathbb{R} is a point or an open interval.
- A cell in \mathbb{R}^{k+1} has one of the two forms

$$\begin{aligned} & \{(a_1, \dots, a_k, a_{k+1}) : (a_1, \dots, a_k) \in C_k \wedge a_{k+1} = r(a_1, \dots, a_k)\} \\ & \{(a_1, \dots, a_k, a_{k+1}) : (a_1, \dots, a_k) \in C_k \wedge r_1(a_1, \dots, a_k) < a_{k+1} < r_2(a_1, \dots, a_k)\} \end{aligned}$$

where C_k is a cell in \mathbb{R}^k , r is a continuous algebraic function, and r_1 and r_2 are continuous algebraic functions, $-\infty$, or ∞ , and $r_1 < r_2$ on C_k .

The Cylindrical Algebraic Decomposition (CAD) algorithm [7, 6, 33] can be used to compute a cell decomposition of any semialgebraic set presented by a real polynomial system. An alternative method of computing cell decompositions is given in [12]. Cell decompositions computed by the CAD algorithm can be represented directly [4, 33] as cylindrical algebraic formulas (CAF; a precise definition is given in Section 2).

Example 1. The following formula $F(x, y, z)$ is a CAF representation of a cell decomposition of the closed unit ball.

$$\begin{aligned} F(x, y, z) &:= (x = -1 \wedge y = 0 \wedge z = 0) \vee (-1 < x < 1 \wedge b_2(x, y, z)) \vee \\ & \quad (x = 1 \wedge y = 0 \wedge z = 0) \\ b_2(x, y, z) &:= (y = R_1(x) \wedge z = 0) \vee (R_1(x) < y < R_2(x) \wedge b_{2,2}(x, y, z)) \vee \\ & \quad (y = R_2(x) \wedge z = 0) \\ b_{2,2}(x, y, z) &:= z = R_3(x, y) \vee R_3(x, y) < z < R_4(x, y) \vee z = R_4(x, y) \end{aligned}$$

where

$$\begin{aligned} R_1(x) &= \text{Root}_{y,1}(x^2 + y^2) = -\sqrt{1 - x^2} \\ R_2(x) &= \text{Root}_{y,2}(x^2 + y^2) = \sqrt{1 - x^2} \\ R_3(x, y) &= \text{Root}_{z,1}(x^2 + y^2 + z^2) = -\sqrt{1 - x^2 - y^2} \\ R_4(x, y) &= \text{Root}_{z,2}(x^2 + y^2 + z^2) = \sqrt{1 - x^2 - y^2} \end{aligned}$$

The CAF representation of a semialgebraic set A can be used to decide whether A is nonempty, to find the minimal and maximal values of the first coordinate of elements of A , to generate an arbitrary element of A , to find a graphical representation of A , to compute the volume of A or to compute multidimensional integrals over A (see [31]).

In our ISSAC conference paper [34] we presented an algorithm, *CAFCombine*, computing Boolean operations on cylindrical algebraic formulas. In this extended version of the paper we investigate how *CAFCombine* can be used to construct a divide-and-conquer algorithm for computing a cylindrical algebraic decomposition. The divide-and-conquer algorithm depends on the algorithm *Subdivide*. Given a real polynomial system $S(x_1, \dots, x_n)$, *Subdivide* finds a Boolean formula Φ and real polynomial systems P_1, \dots, P_m such that

$$S(x_1, \dots, x_n) \Leftrightarrow \Phi(P_1(x_1, \dots, x_n), \dots, P_m(x_1, \dots, x_n))$$

Algorithm 2. (*DivideAndConquerCAD*)

Input: A real polynomial system $S(x_1, \dots, x_n)$.

Output: A cylindrical algebraic formula $F(x_1, \dots, x_n)$ equivalent to $S(x_1, \dots, x_n)$.

- (1) Use *Subdivide* to find Φ and P_1, \dots, P_m such that

$$S \Leftrightarrow \Phi(P_1, \dots, P_m)$$

- (2) If $m = 1$ and $P_1 = S$ then return $\text{CAD}(S)$.

- (3) For $1 \leq i \leq m$, compute

$$F_i := \text{CAD}(P_i)$$

- (4) Use *CAFCombine* to compute a CAF F equivalent to

$$\Phi(F_1, \dots, F_m)$$

- (5) Return F .

The practical usefulness of *DivideAndConquerCAD* depends on the choice of the algorithm *Subdivide*. Our experiments suggest that *DivideAndConquerCAD* is likely to be faster than a direct CAD computation if Φ is a disjunction and, for any for $i \neq j$, P_i and P_j contain few polynomials in common. We propose an algorithm *Subdivide* for polynomial systems S given in disjunctive normal form. The algorithm is based on the connectivity structure of a graph whose vertices are the disjunction terms of S and whose edges depend on polynomials shared between the disjunction terms of S .

Example 3. Let $S(x, y)$ be the result of eliminating the quantifier from

$$\begin{aligned} S_0(x, y) &:= \exists z \quad (z^2(-151x - 740y - 642) + z(-39x + 285y - 634) - 241x - \\ &\quad 57y - 985 < 0 \wedge z^2(275x - 144y + 128) + z(94x - 658y - 267) + \\ &\quad 973x - 810y + 928 = 0 \wedge z^2(-310x - 224y + 144) + \\ &\quad z(-256x - 143y - 77) + 945x - 260y + 825 \leq 0) \end{aligned}$$

using the virtual term substitution algorithm ([36], *Mathematica* command *Resolve*[S_0 , *Reals*]). $S(x, y)$ is a disjunction of 43 conjunctions of polynomial equations and inequalities.

Problem: Find a cell decomposition for the solution set of $S(x, y)$.

Method 1: A direct application of the CAD algorithm. The computation takes 48 seconds.

Method 2: *DivideAndConquerCAD* with *Subdivide* which represents $S(x, y)$ as a disjunction of 43 subformulas, each subformula equal to one of the conjunctions in $S(x, y)$. The computation takes 8.5 seconds.

Method 3: *DivideAndConquerCAD* with graph-based *Subdivide* which represents $S(x, y)$ as a disjunction of 5 subformulas. The subformulas are disjunctions of, respectively, 24, 11, 6, 1, and 1 of the conjunctions in $S(x, y)$. The computation takes 2.4 seconds.

The paper is organized as follows. Section 2 defines cylindrical algebraic formulas. The algorithms *CAFCombine* and *Subdivide* are presented in sections 3 and 4. The last section contains experimental data comparing the performance of *DivideAndConquerCAD* and of direct CAD computation.

2. CYLINDRICAL ALGEBRAIC FORMULAS

Definition 4. A real algebraic function given by defining polynomial $f \in \mathbb{Z}[x_1, \dots, x_n, y]$ and root number $p \in \mathbb{N}_+$ is the function

$$(2.1) \quad \text{Root}_{y,p}f : \mathbb{R}^n \ni (x_1, \dots, x_n) \longrightarrow \text{Root}_{y,p}f(x_1, \dots, x_n) \in \mathbb{R}$$

where $\text{Root}_{y,p}f(x_1, \dots, x_n)$ is the p -th real root of f treated as a univariate polynomial in y . The function is defined for those values of x_1, \dots, x_n for which $f(x_1, \dots, x_n, y)$ has at least p real roots. The real roots are ordered by the increasing value, counting multiplicities. A real algebraic number $\text{Root}_{y,p}f \in \mathbb{R}$ given by defining polynomial $f \in \mathbb{Z}[y]$ and root number p is the p -th real root of f . Let Alg be the set of real algebraic numbers and for $C \subseteq \mathbb{R}^n$ let Alg_C denote the set of all algebraic functions defined and continuous on C . (See [28, 31] for more details on how algebraic numbers and functions can be implemented in a computer algebra system.)

Definition 5. A set $P \subseteq \mathbb{R}[x_1, \dots, x_n, y]$ is *delineable* over $C \subseteq \mathbb{R}^n$ iff

- (1) $\forall f \in P \exists k_f \in \mathbb{N} \forall a \in C \# \{b \in \mathbb{R} : f(a, b) = 0\} = k_f$.
- (2) For any $f \in P$ and $1 \leq p \leq k_f$, $\text{Root}_{y,p}f$ is a continuous function on C .
- (3)

$$\begin{aligned} \forall f, g \in P \quad & (\exists a \in C \text{Root}_{y,p}f(a) = \text{Root}_{y,q}g(a) \Leftrightarrow \\ & \forall a \in C \text{Root}_{y,p}f(a) = \text{Root}_{y,q}g(a)) \end{aligned}$$

Definition 6. A cylindrical system of algebraic constraints in variables x_1, \dots, x_n is a sequence $A = (A_1, \dots, A_n)$ satisfying the following conditions.

- (1) For $1 \leq k \leq n$, A_k is a set of formulas

$$A_k = \{a_{i_1, \dots, i_k}(x_1, \dots, x_k) : 1 \leq i_1 \leq m \wedge 1 \leq i_2 \leq m_{i_1} \wedge \dots \wedge 1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}\}$$

- (2) For each $1 \leq i_1 \leq m$, $a_{i_1}(x_1)$ is *true* or

$$x_1 = r$$

where $r \in \text{Alg}$, or

$$r_1 < x_1 < r_2$$

where $r_1 \in \text{Alg} \cup \{-\infty\}$, $r_2 \in \text{Alg} \cup \{\infty\}$ and $r_1 < r_2$. Moreover, if $s_1, s_2 \in \text{Alg} \cup \{-\infty, \infty\}$, s_1 appears in $a_u(x_1)$, s_2 appears in $a_v(x_1)$ and $u < v$ then $s_1 \leq s_2$.

- (3) Let $k < n$, $I = (i_1, \dots, i_k)$ and let $C_I \subseteq \mathbb{R}^k$ be the solution set of

$$(2.2) \quad a_{i_1}(x_1) \wedge a_{i_1, i_2}(x_1, x_2) \wedge \dots \wedge a_{i_1, \dots, i_k}(x_1, \dots, x_k)$$

(a) For each $1 \leq i_{k+1} \leq m_I$,

$$a_{i_1, \dots, i_k, i_{k+1}}(x_1, \dots, x_k, x_{k+1})$$

is *true* or

$$(2.3) \quad x_{k+1} = r(x_1, \dots, x_k)$$

and $r \in \text{Alg}_{C_I}$, or

$$(2.4) \quad r_1(x_1, \dots, x_k) < x_{k+1} < r_2(x_1, \dots, x_k)$$

where $r_1 \in \text{Alg}_{C_I} \cup \{-\infty\}$, $r_2 \in \text{Alg}_{C_I} \cup \{\infty\}$ and $r_1 < r_2$ on C_I .

(b) If $s_1, s_2 \in \text{Alg}_{C_I} \cup \{-\infty, \infty\}$, s_1 appears in

$$a_{i_1, \dots, i_k, u}(x_1)$$

s_2 appears in

$$a_{i_1, \dots, i_k, v}(x_1)$$

and $u < v$ then $s_1 \leq s_2$ on C_I .

(c) Let $P_I \subseteq \mathbb{Z}[x_1, \dots, x_k, x_{k+1}]$ be the set of defining polynomials of all real algebraic functions that appear in formulas a_J for $J = (i_1, \dots, i_k, i_{k+1})$, $1 \leq i_{k+1} \leq m_I$. Then P_I is delineable over C_I .

Definition 7. Let A be a cylindrical system of algebraic constraints in variables x_1, \dots, x_n . Define

$$b_{i_1, \dots, i_n}(x_1, \dots, x_n) := \text{true}$$

For $2 \leq k \leq n$, level k cylindrical algebraic subformulas given by A are the formulas

$$b_{i_1, \dots, i_{k-1}}(x_1, \dots, x_n) := \bigvee_{1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}} a_{i_1, \dots, i_k}(x_1, \dots, x_k) \wedge b_{i_1, \dots, i_k}(x_1, \dots, x_n)$$

The *support cell* of $b_{i_1, \dots, i_{k-1}}$ is the solution set

$$C_{i_1, \dots, i_{k-1}} \subseteq \mathbb{R}^k$$

of

$$a_{i_1}(x_1) \wedge a_{i_1, i_2}(x_1, x_2) \wedge \dots \wedge a_{i_1, \dots, i_{k-1}}(x_1, \dots, x_{k-1})$$

The *cylindrical algebraic formula (CAF)* given by A is the formula

$$F(x_1, \dots, x_n) := \bigvee_{1 \leq i_1 \leq m} a_{i_1}(x_1) \wedge b_{i_1}(x_1, \dots, x_n)$$

Remark 8. Let $F(x_1, \dots, x_n)$ be a CAF given by a cylindrical system of algebraic constraints A . Then

- (1) For $1 \leq k \leq n$, sets C_{i_1, \dots, i_k} are cells in \mathbb{R}^k .
- (2) Cells

$$\{C_{i_1, \dots, i_n} : 1 \leq i_1 \leq m \wedge 1 \leq i_2 \leq m_{i_1} \wedge \dots \wedge 1 \leq i_n \leq m_{i_1, \dots, i_{n-1}}\}$$

form a decomposition of the solution set S_F of F , i.e. they are disjoint and their union is equal to S_F .

Proof. Both parts of the remark follow from the definitions of A and F . \square

Remark 9. Given a real polynomial system $S(x_1, \dots, x_n)$ a version of the CAD algorithm can be used to find a CAF $F(x_1, \dots, x_n)$ equivalent to $S(x_1, \dots, x_n)$.

Proof. The version of CAD described in [33] returns a CAF equivalent to the input system. \square

3. ALGORITHM CAFCOMBINE

In this section we describe the algorithm *CAFCombine*. The algorithm is a modified version of the CAD algorithm. We describe only the modification. For details of the CAD algorithm see [6, 7]. Our implementation is based on the version of CAD described in [33].

Definition 10. Let $P \subseteq \mathbb{R}[x_1, \dots, x_n]$ be a finite set of polynomials and let \bar{P} be the set of irreducible factors of elements of P . $W = (W_1, \dots, W_n)$ is a *projection sequence* for P iff

- (1) *Projection sets* W_1, \dots, W_n are finite sets of irreducible polynomials.
- (2) For $1 \leq k \leq n$, $\bar{P} \cap (\mathbb{R}[x_1, \dots, x_k] \setminus \mathbb{R}[x_1, \dots, x_{k-1}]) \subseteq W_k \subseteq \mathbb{R}[x_1, \dots, x_k] \setminus \mathbb{R}[x_1, \dots, x_{k-1}]$.
- (3) If $k < n$ and all polynomials of W_k have constant signs on a cell $C \subseteq \mathbb{R}^k$, then all polynomials of W_{k+1} that are not identically zero on $C \times \mathbb{R}$ are delineable over C .

Remark 11. For an arbitrary finite set $P \subseteq \mathbb{R}[x_1, \dots, x_n]$ a projection sequence can be computed using Hong's projection operator [14]. McCallum's projection operator [23, 24] gives smaller projection sets for well-oriented sets P .

If $P \subseteq Q \subseteq \mathbb{R}[x_1, \dots, x_n]$ and W is a projection sequence for Q then W is a projection sequence for P .

Notation 12. For a CAF F , let P_F denote the set of defining polynomials of all algebraic numbers and functions that appear in F .

First let us prove the following rather technical effective lemmas that will be used in the algorithm. We use notation of Definition 7.

Lemma 13. *Let*

$$F(x_1, \dots, x_n) := \bigvee_{1 \leq i_1 \leq m} a_{i_1}(x_1) \wedge b_{i_1}(x_1, \dots, x_n)$$

be a CAF and let $-\infty = r_0 < r_1 < \dots < r_l < r_{l+1} = \infty$ be such that all real roots of elements of $P_F \cap \mathbb{R}[x_1]$ are among r_1, \dots, r_l . Let $a(x_1)$ be either $x_1 = r_j$ for some $1 \leq j \leq l$, or $r_j < x_1 < r_{j+1}$ for some $0 \leq j \leq l$, and let C_a be the solution set of a . Then

$$(3.1) \quad \forall x_1 \in C_a \quad F(x_1, \dots, x_n) \Leftrightarrow G(x_1, \dots, x_n)$$

and one of the following two statements is true

- (1) *There exist $1 \leq i_1 \leq m$ such that $C_a \subseteq C_{i_1}$ and $G(x_1, \dots, x_n) = b_{i_1}(x_1, \dots, x_n)$.*
- (2) *For all $1 \leq i_1 \leq m$, $C_a \cap C_{i_1} = \emptyset$ and $G(x_1, \dots, x_n) = \text{false}$*

Moreover, given F and a , G can be found algorithmically.

Proof. Let r be an algebraic number that appears in a_{i_1} . Then $r = \text{Root}_{x_1, p} f$ for some $f \in P_F$. Hence, $r = r_{j_0}$ for some $1 \leq j_0 \leq l$ and the value of j_0 can be determined algorithmically. If a is $x_1 = r_j$, then $C_a \cap C_{i_1} \neq \emptyset$ iff a_{i_1} is either $x_1 = r_j$ or $r_u < x_1 < r_v$ with $u < j < v$. In both cases $C_a \subseteq C_{i_1}$. If a is $r_j < x_1 < r_{j+1}$, then $C_a \cap C_{i_1} \neq \emptyset$ iff a_{i_1} is $r_u < x_1 < r_v$ with $u \leq j$ and $v \geq j+1$. In this case also $C_a \subseteq C_{i_1}$. Equivalence (3.1) follows from the statements (1) and (2). \square

Lemma 14. *Let $2 \leq k \leq n$, let*

$$b_{i_1, \dots, i_{k-1}}(x_1, \dots, x_n) := \bigvee_{1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}} a_{i_1, \dots, i_k}(x_1, \dots, x_k) \wedge b_{i_1, \dots, i_k}(x_1, \dots, x_n)$$

be a level k cylindrical algebraic subformula of a CAF F , let $W = (W_1, \dots, W_n)$ be a projection sequence for P_F . Let $C \subseteq \mathbb{R}^{k-1}$ be a cell such that all polynomials of W_{k-1} have constant signs on C and $C \subseteq C_{i_1, \dots, i_{k-1}}$. Let $(c_1, \dots, c_{k-1}) \in C$ and let $d_1 < \dots < d_l$ be all real roots of $\{f(c_1, \dots, c_{k-1}, x_k) : f \in W_k\}$. For $1 \leq j \leq l$, let $r_j := \text{Root}_{x_k, p} f$, where $f \in W_k$

and d_j is the p -th root of $f(c_1, \dots, c_{k-1}, x_k)$. Let $a(x_1, \dots, x_k)$ be either $x_k = r_j$ for some $1 \leq j \leq l$, or $r_j < x_k < r_{j+1}$ for some $0 \leq j \leq l$, where $r_0 := -\infty$ and $r_{l+1} := \infty$ and let

$$C_a := \{(x_1, \dots, x_k) : (x_1, \dots, x_{k-1}) \in C \wedge a(x_1, \dots, x_k)\}$$

Then

$$(3.2) \quad \forall (x_1, \dots, x_k) \in C_a \quad b_{i_1, \dots, i_{k-1}}(x_1, \dots, x_n) \Leftrightarrow G(x_1, \dots, x_n)$$

and one of the following two statements is true

- (1) There exist $1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}$ such that

$$C_a \subseteq C_{i_1, \dots, i_{k-1}, i_k}$$

and

$$G(x_1, \dots, x_n) = b_{i_1, \dots, i_k}(x_1, \dots, x_n)$$

- (2) For all $1 \leq i_k \leq m_{i_1, \dots, i_{k-1}}$

$$C_a \cap C_{i_1, \dots, i_{k-1}, i_k} = \emptyset$$

and

$$G(x_1, \dots, x_n) = \text{false}$$

Moreover, given $b_{i_1, \dots, i_{k-1}}$, a , (c_1, \dots, c_{k-1}) , d_1, \dots, d_l and the multiplicity of d_j as a root of f , for all $1 \leq j \leq l$ and $f \in W_k$, G can be found algorithmically.

Proof. Let r be an algebraic function that appears in a_{i_1, \dots, i_k} . Then $r = \text{Root}_{x_k, p} f$ for some $f \in P_F$. By Definition 6, r is defined and continuous on C . Since W is a projection sequence for P_F , all factors of f that depend on x_k are elements of W_k . Hence, $r(c_1, \dots, c_{k-1}) = d_{j_0}$ for some $1 \leq j_0 \leq l$. Since d_{j_0} is the p -th of real roots of factors of f , multiplicities counted, if the multiplicity of d_j as a root of f is known for all $1 \leq j \leq l$ and $f \in W_k$, the value of j_0 can be determined algorithmically. Since all polynomials of W_{k-1} have constant signs on C , all elements of W_k that are not identically zero on C are delineable over C . Therefore, $r = r_{j_0}$ and $r_1 < \dots < r_l$ on C . If a is $x_k = r_j$, then $C_a \cap C_{i_1, \dots, i_{k-1}, i_k} \neq \emptyset$ iff a_{i_1, \dots, i_k} is either $x_k = r_j$ or $r_u < x_k < r_v$ with $u < j < v$. In both cases $C_a \subseteq C_{i_1, \dots, i_{k-1}, i_k}$. If a is $r_j < x_k < r_{j+1}$, then $C_a \cap C_{i_1, \dots, i_{k-1}, i_k} \neq \emptyset$ iff a_{i_1, \dots, i_k} is $r_u < x_k < r_v$ with $u \leq j$ and $v \geq j+1$. In this case also $C_a \subseteq C_{i_1, \dots, i_{k-1}, i_k}$. Equivalence (3.2) follows from the statements (1) and (2). \square

Let us now describe two subalgorithms used in *CAFCombine*. The first, recursive, subalgorithm requires its input to satisfy the following conditions.

- (1) $W = (W_1, \dots, W_n)$ is a projection sequence for $P_{F_1} \cup \dots \cup P_{F_m}$, where

$$F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n)$$

are cylindrical algebraic formulas.

- (2) $(c_1, \dots, c_{k-1}) \in C$, $2 \leq k \leq n$ and $C \subseteq \mathbb{R}^{k-1}$ is a cell such that all polynomials of W_{k-1} have constant signs on C .
- (3) Each B_j is a level k cylindrical algebraic formula of F_j or *false*.
- (4) C is contained in the intersection of support cells of all B_j that are not *false*.
- (5) $\Phi(p_1, \dots, p_m)$ is a Boolean formula.

Algorithm 15. (*Lift*)

Input: $(c_1, \dots, c_{k-1}) \in \mathbb{R}^{k-1}$, W , B_1, \dots, B_m , Φ .

Output: A level k cylindrical algebraic subformula

$$b(x_1, \dots, x_n)$$

such that

$$(3.3) \quad \forall (x_1, \dots, x_{k-1}) \in Cb(x_1, \dots, x_n) \Leftrightarrow \Phi(B_1(x_1, \dots, x_n), \dots, B_m(x_1, \dots, x_n))$$

- (1) Let $d_1 < \dots < d_l$ be all real roots of

$$\{f(c_1, \dots, c_{k-1}, x_k) : f \in W_k\}$$

- (2) For $1 \leq i \leq l$, let $r_i := \text{Root}_{x_k, p} f$, where $f \in W_k$ and d_i is the p -th root of $f(c_1, \dots, c_{k-1}, x_k)$.

- (3) For $f \in W_k$, if d_i is a root of $f(c_1, \dots, c_{k-1}, x_k)$, let $M(f, i)$ be its multiplicity, otherwise $M(f, i) := 0$.

- (4) For $1 \leq i \leq l$, set $a_{2i}(x_1, \dots, x_k) := (x_k = r_i)$ and $c_{k, 2i} := d_i$.

- (5) For $0 \leq i \leq l$, set

$$a_{2i+1}(x_1, \dots, x_k) := (r_i < x_k < r_{i+1})$$

where $r_0 := -\infty$ and $r_{l+1} := \infty$, and pick $c_{k, 2i+1} \in (d_i, d_{i+1}) \cap \mathbb{Q}$, where $d_0 := -\infty$ and $d_{l+1} := \infty$.

- (6) For $1 \leq i \leq 2l+1$

- (a) For $1 \leq j \leq m$, if $B_j = \text{false}$, set $G_j = \text{false}$, otherwise let G_j be the formula G found using Lemma 14 applied to B_j , a_i , (c_1, \dots, c_{k-1}) , d_1, \dots, d_l and M .

- (b) Let $\Psi := \Phi(G_1, \dots, G_m)$. If Ψ is true or false, set $b_i(x_1, \dots, x_n) := \Psi$.

- (c) Otherwise set $b_i(x_1, \dots, x_n)$ to

$$\text{Lift}((c_1, \dots, c_{k-1}, c_{k, i}), W, G_1, \dots, G_m, \Phi)$$

- (7) Return

$$b(x_1, \dots, x_n) := \bigvee_{1 \leq i \leq 2l+1} a_i(x_1, \dots, x_k) \wedge b_i(x_1, \dots, x_n)$$

The second subalgorithm requires its input to satisfy the following conditions.

- (1) Either $a(x_1) = (x_1 = r)$, where $r \in \text{Alg}$, or $a(x_1) = (r < x_1 < s)$, where $r \in \text{Alg} \cup \{-\infty\}$ and $s \in \text{Alg} \cup \{\infty\}$.
- (2) For $1 \leq j \leq m$, B_j is a level 2 cylindrical algebraic subformula of a CAF

$$F_j(x_1, \dots, x_n) := a(x_1) \wedge B_j(x_1, \dots, x_n)$$

- (3) $\Phi(p_1, \dots, p_m)$ is a Boolean formula.

Algorithm 16. (*CombineStacks*)

Input: a, B_1, \dots, B_m, Φ .

Output: A CAF $F(x_1, \dots, x_n)$ such that

$$(3.4) \quad F(x_1, \dots, x_n) \iff \Phi(F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n))$$

- (1) Let $W := (W_1, \dots, W_n)$ be a projection sequence for $P_{F_1} \cup \dots \cup P_{F_m}$.

- (2) If $a(x_1) = (x_1 = r)$ then set $l := 0$, $a_1(x_1) := (x_1 = r)$, and $c_{1, 1} := r$.

- (3) If $a(x_1) = (r < x_1 < s)$ then

- (a) Let $r_1 < \dots < r_l$ be all real roots of elements of W_1 in (r, s) .

- (b) For $1 \leq i \leq l$, set $a_{2i}(x_1) := (x_1 = r_i)$ and $c_{1, 2i} := r_i$.

- (c) For $0 \leq i \leq l$, set $a_{2i+1}(x_1) := (r_i < x_1 < r_{i+1})$ and pick $c_{1, 2i+1} \in (r_i, r_{i+1}) \cap \mathbb{Q}$, where $r_0 := r$ and $r_{l+1} := s$.

- (4) For $1 \leq i \leq 2l+1$ set

$$b_i(x_1, \dots, x_n) := \text{Lift}((c_{1, i}), W, B_1, \dots, B_m, \Phi)$$

(5) *Return*

$$F(x_1, \dots, x_n) := \bigvee_{1 \leq i \leq 2l+1} a_i(x_1) \wedge b_i(x_1, \dots, x_n)$$

We can now describe the algorithm *CADCombine* (cf. [34], Algorithm 17).

Algorithm 17. (*CAFCombine*)

Input: Cylindrical algebraic formulas

$$F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n)$$

and a Boolean formula $\Phi(p_1, \dots, p_m)$.

Output: A CAF $F(x_1, \dots, x_n)$ such that

$$(3.5) \quad F(x_1, \dots, x_n) \iff \Phi(F_1(x_1, \dots, x_n), \dots, F_m(x_1, \dots, x_n))$$

- (1) Let $r_1 < \dots < r_l$ be all real roots of $(P_{F_1} \cup \dots \cup P_{F_m}) \cap \mathbb{R}[x_1]$.
- (2) For $1 \leq i \leq l$, set $a_{2i}(x_1) := (x_1 = r_i)$ and for $0 \leq i \leq l$, set $a_{2i+1}(x_1) := (r_i < x_1 < r_{i+1})$, where $r_0 := -\infty$ and $r_{l+1} := \infty$.
- (3) For $1 \leq i \leq 2l+1$
 - (a) For $1 \leq j \leq m$, let G_j be the formula G found using Lemma 13 applied to F_j and a_i .
 - (b) Let j_1, \dots, j_s be all $1 \leq j \leq m$ for which G_j is neither true nor false.
 - (c) Let $\Psi(p_{j_1}, \dots, p_{j_s})$ be the formula obtained from Φ by replacing p_j with G_j for all j for which G_j is true or false.
 - (d) If Ψ is true or false, set $H_i(x_1, \dots, x_n) := a_i \wedge \Psi$.
 - (e) Otherwise set $H_i(x_1, \dots, x_n)$ to

$$\text{CombineStacks}(a_i; G_{j_1}, \dots, G_{j_s}; \Psi)$$

- (4) *Return* $F(x_1, \dots, x_n) := \bigvee_{1 \leq i \leq 2l+1} H_i(x_1, \dots, x_n)$.

Proof. (Correctness of the algorithms) To show correctness of *CombineStacks*, let us first show that inputs to *Lift* satisfy the required conditions. Condition (1) follows from step 1 of *CombineStacks*. If $k = 2$, the cell C is defined as a root or the open interval between two subsequent roots of polynomials of W_1 . For $k > 2$, the cell C is defined as a graph of a root or the set between graphs of two subsequent roots of polynomials of W_{k-1} over a cell on which W_{k-1} is delineable. This proves condition (2). Conditions (3) and (4) are guaranteed by Lemmas 13 and 14. Finally, (5) is satisfied, because Φ is always the same formula, given as input to *CombineStacks*.

To complete the proof we need to show the equivalences (3.4) and (3.3). Equivalence (3.3) follows from Lemma 14 and the fact that the sets

$$\{(x_1, \dots, x_k) : (x_1, \dots, x_{k-1}) \in C \wedge a_i(x_1, \dots, x_k)\}$$

are disjoint and their union is equal to $C \times \mathbb{R}$. Equivalence (3.4) follows from Lemma 13 and the fact that the sets $\{x_1 \in \mathbb{R} : a_i(x_1)\}$ are disjoint and their union is equal to \mathbb{R} .

Correctness of *CAFCombine* follows from Lemma 13, correctness of *CombineStacks*, and the fact that the sets $\{x_1 \in \mathbb{R} : a_i(x_1)\}$ are disjoint and their union is equal to \mathbb{R} . \square

Example 18. Let

$$\begin{aligned} f_1 &:= (x+1)^4 + y^4 - 4 \\ g_1 &:= (x+2)^2 + y^2 - 5 \\ f_2 &:= (x-1)^4 + y^4 - 4 \\ g_2 &:= (x-2)^2 + y^2 - 5 \end{aligned}$$

and let

$$A_1 := \{(x, y) \in \mathbb{R}^2 : f_1 < 0 \wedge g_1 < 0\}$$

$$A_2 := \{(x, y) \in \mathbb{R}^2 : f_2 < 0 \wedge g_2 < 0\}$$

The following CAFs represent cell decompositions of A_1 and A_2 .

$$\begin{aligned} F_1(x, y) &:= r_1 < x < r_2 \wedge \text{Root}_{y,1}f_1 < y < \text{Root}_{y,2}f_1 \vee \\ &\quad x = r_2 \wedge \text{Root}_{y,1}f_1 < y < \text{Root}_{y,2}f_1 \vee \\ &\quad r_2 < x < r_4 \wedge \text{Root}_{y,1}g_1 < y < \text{Root}_{y,2}g_1 \\ F_2(x, y) &:= r_3 < x < r_5 \wedge \text{Root}_{y,1}g_2 < y < \text{Root}_{y,2}g_2 \vee \\ &\quad x = r_5 \wedge \text{Root}_{y,1}g_2 < y < \text{Root}_{y,2}g_2 \vee \\ &\quad r_5 < x < r_6 \wedge \text{Root}_{y,1}f_2 < y < \text{Root}_{y,2}f_2 \end{aligned}$$

where

$$\begin{aligned} r_1 &:= -1 - \sqrt{2} \approx -2.414 \\ r_2 &:= \text{Root}_{x,1}x^4 + 6x^3 + 10x^2 - 2x - 1 \approx -0.244 \\ r_3 &:= 2 - \sqrt{5} \approx -0.236 \\ r_4 &:= -2 + \sqrt{5} \approx 0.236 \\ r_5 &:= \text{Root}_{x,2}x^4 - 6x^3 + 10x^2 + 2x - 1 \approx 0.244 \\ r_6 &:= 1 + \sqrt{2} \approx 2.414 \end{aligned}$$

Compute a CAF representation of $A_1 \cap A_2$ (Figure 1) using *CAFCombine*.

The input consists of F_1 , F_2 and $\Phi(p_1, p_2) := p_1 \wedge p_2$. The roots computed in step (1) are r_1, r_2, r_3, r_4, r_5 and r_6 . In step (3), for all $i \neq 7$, either G_1 or G_2 is *false*, and hence $\Psi = \text{false}$. For $i = 7$ the algorithm computes *CombineStacks*($a_7; G_1, G_2; \Phi$), where

$$\begin{aligned} a_7 &:= r_3 < x < r_4 \\ G_1 &:= \text{Root}_{y,1}g_1 < y < \text{Root}_{y,2}g_1 \\ G_2 &:= \text{Root}_{y,1}g_2 < y < \text{Root}_{y,2}g_2 \end{aligned}$$

The projection sequence computed in step (1) of *CombineStacks* is

$$\begin{aligned} W_2 &:= \{g_1, g_2\} \\ W_1 &:= \{x, x^2 + 4x - 1, x^2 - 4x - 1\} \end{aligned}$$

The only root of W_1 in (r_3, r_4) is 0. The returned cell decomposition of $A_1 \cap A_2$ consists of three cells constructed by *Lift* over cells $r_3 < x < 0$, $x = 0$, and $0 < x < r_4$.

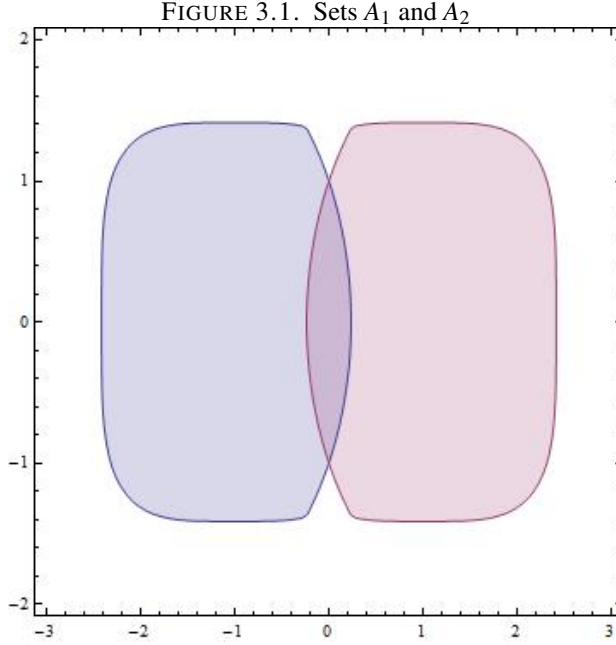
$$\begin{aligned} F(x, y) &:= r_3 < x < 0 \wedge \text{Root}_{y,1}g_2 < y < \text{Root}_{y,2}g_2 \vee \\ &\quad x = 0 \wedge -1 < y < 1 \vee \\ &\quad 0 < x < r_4 \wedge \text{Root}_{y,1}g_1 < y < \text{Root}_{y,2}g_1 \end{aligned}$$

Note that the computation did not require including f_1 and f_2 in the projection set.

4. ALGORITHM SUBDIVIDE

In this section we propose an algorithm for subdividing polynomial systems $S(x_1, \dots, x_n)$ given in disjunctive normal form. From experimenting with various subdivision methods we deduced the following rules for designing a subdivision heuristic.

- Do not subdivide conjunctions.



- Group terms of the disjunction so that different groups have as few common polynomials as possible.
- Common polynomials that contain x_n matter much more than polynomials that do not contain x_n .
- Common polynomials with higher degrees in x_n matter more than polynomials with lower degrees in x_n .

This led us to the following subdivision algorithm. The algorithm depends on a parameter $0 \leq p \leq 1$ to be determined experimentally.

Notation 19. Let $S(x_1, \dots, x_n)$ and $T(x_1, \dots, x_n)$ be real polynomial systems. Let $Wt(S)$ denote the sum of degrees in x_n of all distinct polynomials that appear in S , and let $Wt(S, T)$ denote the sum of degrees in x_n of all distinct polynomials that appear both in S and in T .

Algorithm 20. (*Subdivide*)

Input: A real polynomial system $S(x_1, \dots, x_n)$.

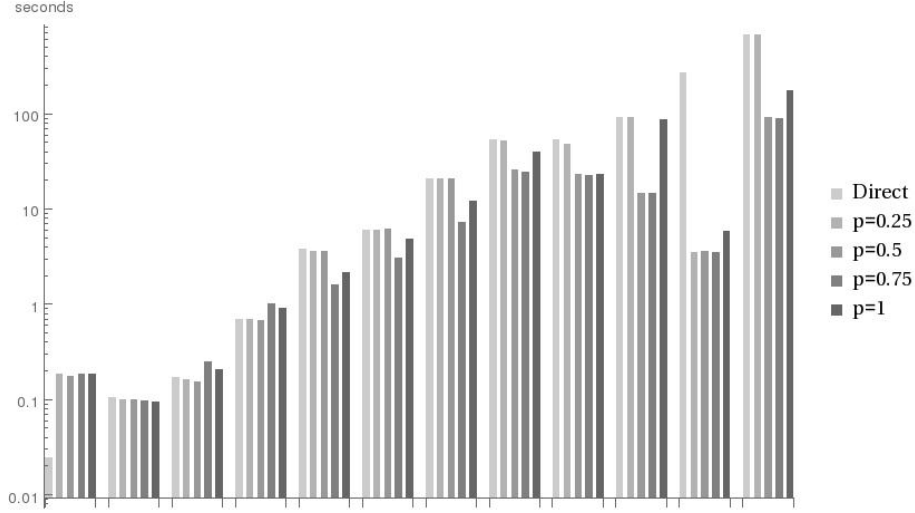
Output: A Boolean formula Φ and real polynomial systems P_1, \dots, P_m such that

$$S(x_1, \dots, x_n) \Leftrightarrow \Phi(P_1(x_1, \dots, x_n), \dots, P_m(x_1, \dots, x_n))$$

- (1) If S is not a disjunction return $\Phi := Id$ and $P_1 := S$.
- (2) Let $S = S_1 \vee \dots \vee S_k$. Construct a graph G as follows.
 - (a) The vertices of G are S_1, \dots, S_k .
 - (b) There is an edge connecting S_i and S_j if $Wt(S_i, S_j) > 0$ and

$$Wt(S_i, S_j) \geq p \min(Wt(S_i), Wt(S_j))$$
- (3) Compute the connected components $\{S_{1,1}, \dots, S_{1,l_1}\}, \dots, \{S_{m,1}, \dots, S_{m,l_m}\}$ of G .
- (4) For $1 \leq j \leq m$ set $P_j := S_{j,1} \vee \dots \vee S_{j,l_j}$.
- (5) Set $\Phi(p_1, \dots, p_m) := p_1 \vee \dots \vee p_m$.
- (6) Return Φ and P_1, \dots, P_m .

FIGURE 5.1. Examples from [5]



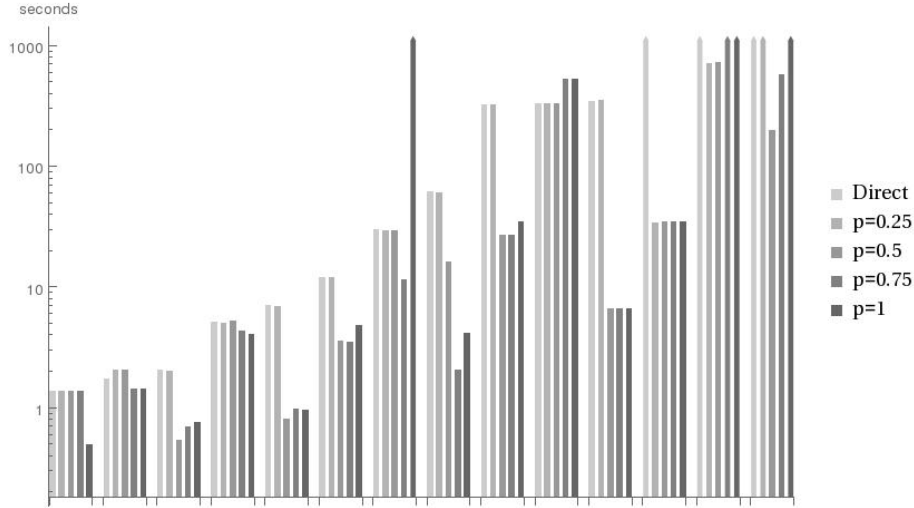
5. EMPIRICAL RESULTS

In this section we compare performance of *DivideAndConquerCAD* and of direct CAD computation. As benchmark problems we chose formulas obtained by application of virtual term substitution methods to quantifier elimination problems, because such formulas are “naturally occurring” CAD inputs that are disjunctions of many terms. We ran four variants of *DivideAndConquerCAD*, corresponding to different choices of the parameter p in *Subdivide*. We used $p = 0.25$, $p = 0.5$, $p = 0.75$, and $p = 1$. The algorithms have been implemented in C, as a part of the kernel of *Mathematica*. For direct CAD computation the algorithms use the *Mathematica* implementation of the version of CAD described in [33]. The experiments have been conducted on a Linux virtual machine with a 3.07 GHz Intel Core i7 processor and 6 GB of RAM available. Each computation was given a time limit of 1200 seconds.

5.1. Examples from [5]. In [5] there are 56 examples obtained by application of virtual term substitution methods to quantifier elimination problems. Here we used 28 of the examples, 7 from applications and 21 randomly generated, for which there were between 2 and 4 free variables. In 22 of the 28 examples at least one method finished within the time limit and the difference between the slowest and the fastest timing was at least 10%. In 10 of the 22 examples the input systems were subdivided only for $p = 1$, and *DivideAndConquerCAD* with $p = 1$ was slower than direct CAD computation. Timings for the remaining 12 examples are shown in Figure 5.1 (note the logarithmic scale). In 9 of the 12 examples *DivideAndConquerCAD* with $p = 0.75$ is faster than direct CAD computation, in 3 examples it is slower. In the 12 examples *DivideAndConquerCAD* with $p = 0.75$ is the fastest method on average, 2.25 times faster than direct CAD computation.

5.2. Random examples. We generated 16 random examples with 2 or 3 variables. The examples were obtained by elimination of up to three quantifiers using virtual term substitution (with intermediate formula simplification). The initial quantified systems were randomly generated quantified conjunctions of 2-4 polynomial equations or inequalities.

FIGURE 5.2. Random examples



The polynomials were linear in all quantified variables except for the first one and quadratic in the remaining variables. The quadratic term in the first quantifier variable did not contain other quantifier variables. The results of quantifier elimination were put in disjunctive normal form and only disjunctions of at least 10 terms were selected. In 2 examples all timings were the same. Timings for the remaining 14 examples are shown in Figure 5.2 (note the logarithmic scale). In 12 of the 14 examples *DivideAndConquerCAD* with $p = 0.75$ is faster than direct CAD computation, in one example it is slower. In the 14 examples *DivideAndConquerCAD* with $p = 0.75$ is the fastest method on average, at least 3.96 times faster than direct CAD computation (in two examples direct CAD computation did not finish in 1200 seconds and *DivideAndConquerCAD* with $p = 0.75$ did).

5.3. Conclusions. The experiments show that *DivideAndConquerCAD* with graph-based *Subdivide* is often, but not always, faster than direct CAD computation. On average, the best performance was obtained by choosing the parameter value $p = 0.75$ in *Subdivide*.

REFERENCES

- [1] H. Anai, K. Yokoyama, “CAD via Numerical Computation with Validated Symbolic Reconstruction, A3L 2005 Proceedings, 2005, 25-30.
- [2] C. W. Brown, “Improved Projection for Cylindrical Algebraic Decomposition”, J. Symbolic Comp., 32 (2001), 447-465.
- [3] C. W. Brown, “An Overview of QEPCAD B: a Tool for Real Quantifier Elimination and Formula Simplification”, J. JSSAC, Vol 10, No. 1 (2003), 13-22.
- [4] C. W. Brown, “QEPCAD B - a program for computing with semi-algebraic sets using CADs”, ACM SIGSAM Bulletin, 37 (4), 2003, 97-108.
- [5] C. W. Brown, A. Strzebonski, Black-Box/White-Box Simplification and Applications to Quantifier Elimination, Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2010, 69-76, Munich, Germany, July 25-28, 2010. ACM, Stephen M. Watt, ed.
- [6] B. Caviness, J. Johnson (eds.), “Quantifier Elimination and Cylindrical Algebraic Decomposition”, Springer-Verlag 1998.
- [7] G. E. Collins, “Quantifier Elimination for the Elementary Theory of Real Closed Fields by Cylindrical Algebraic Decomposition”, Lect. Notes Comput. Sci., 33, 1975, 134-183.

- [8] G. E. Collins, "Quantifier Elimination by Cylindrical Algebraic Decomposition - Twenty Years of Progress", in B. Caviness, J. Johnson (eds.), *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Springer-Verlag 1998, 8-23.
- [9] G. E. Collins, "Application of Quantifier Elimination to Solotareff's Approximation Problem", Technical Report 95-31, RISC Report Series, University of Linz, Austria, 1995.
- [10] G. E. Collins, H. Hong, "Partial Cylindrical Algebraic Decomposition for Quantifier Elimination", *J. Symbolic Comp.*, 12 (1991), 299-328.
- [11] G. E. Collins, J. R. Johnson, W. Krandick, "Interval Arithmetic in Cylindrical Algebraic Decomposition", *J. Symbolic Comp.*, 34 (2002), 145-157.
- [12] C. Chen, M. Moreno Maza, B. Xia, L. Yang, "Computing Cylindrical Algebraic Decomposition via Triangular Decomposition", *Proceedings of ISSAC 2009*, 95-102.
- [13] P. Dorato, W. Yang, C. Abdallah, "Robust Multi-Objective Feedback Design by Quantifier Elimination", *J. Symbolic Comp.*, 24 (1997), 153-160.
- [14] H. Hong, "An Improvement of the Projection Operator in Cylindrical Algebraic Decomposition", *Proceedings of ISSAC 1990*, 261-264.
- [15] H. Hong, "Efficient Method for Analyzing Topology of Plane Real Algebraic Curves", *Proceedings of IMACS-SC 93*, Lille, France, 1993.
- [16] H. Hong, R. Liska, S. Steinberg, "Testing Stability by Quantifier Elimination", *J. Symbolic Comp.*, 24 (1997), 161-188.
- [17] M. Jirstrand, "Nonlinear Control System Design by Quantifier Elimination", *J. Symbolic Comp.*, 24 (1997), 137-152.
- [18] M. A. G. Jenkins, "Three-stage variable-shift iterations for the solution of polynomial equations with a posteriori error bounds for the zeros", Ph.D. dissertation, Stanford University, 1969.
- [19] J. B. Keiper, D. Withoff, "Numerical Computation in Mathematica", Course Notes, *Mathematica Conference 1992*.
- [20] D. Lazard, "Solving Kaltoven's Challenge on Zolotarev's Approximation Problem", *Proceedings of ISSAC 2006*, 196-203.
- [21] S. Łojasiewicz, "Ensembles semi-analytiques", I.H.E.S. (1964).
- [22] R. Loos, V. Weispfenning, "Applying Linear Quantifier Elimination", *The Computer Journal*, Vol. 36, No. 5, 1993, 450-461.
- [23] S. McCallum, "An Improved Projection for Cylindrical Algebraic Decomposition of Three Dimensional Space", *J. Symbolic Comp.*, 5 (1988), 141-161.
- [24] S. McCallum, "An Improved Projection for Cylindrical Algebraic Decomposition", in B. Caviness, J. Johnson (eds.), *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Springer-Verlag 1998, 242-268.
- [25] S. McCallum, "On Projection in CAD-Based Quantifier Elimination with Equational Constraint", In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation*, ACM Press 1999, 145-149.
- [26] S. McCallum, "On Propagation of Equational Constraints in CAD-Based Quantifier Elimination", In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, ACM Press 2001, 223-230.
- [27] A. Strzebonski, "An Algorithm for Systems of Strong Polynomial Inequalities", *The Mathematica Journal*, vol. 4, iss. 4 (1994), 74-77.
- [28] A. Strzebonski, "Computing in the Field of Complex Algebraic Numbers", *J. Symbolic Comp.*, 24 (1997), 647-656.
- [29] A. Strzebonski, "Algebraic Numbers in Mathematica 3.0", *The Mathematica Journal*, vol. 6, iss. 4 (1996), 74-80.
- [30] A. Strzebonski, "A Real Polynomial Decision Algorithm Using Arbitrary-Precision Floating Point Arithmetic", *Reliable Computing*, Vol. 5, Iss. 3 (1999), 337-346.
- [31] A. Strzebonski, "Solving Systems of Strict Polynomial Inequalities", *J. Symbolic Comp.*, 29 (2000), 471-480.
- [32] A. Strzebonski, "Solving Algebraic Inequalities", *The Mathematica Journal*, Vol. 7, Iss. 4 (2000), 525-541.
- [33] A. Strzebonski, "Cylindrical Algebraic Decomposition using validated numerics", *J. of Symbolic Comp.* 41 (2006), 1021-1038.
- [34] A. Strzebonski, "Computation with Semialgebraic Sets Represented by Cylindrical Algebraic Formulas", *ISSAC 2010*, 61-68.
- [35] A. Tarski, "A decision method for elementary algebra and geometry", University of California Press, Berkeley 1951.

- [36] V. Weispfenning, "Quantifier elimination for real algebra - the quadratic case and beyond", Appl. Algebra Eng. Commun. Comput., 8 (1997), 85-101.
- [37] S. Wolfram, "The Mathematica Book", 4th. Ed., Wolfram Media/Cambridge University Press, 1999.

WOLFRAM RESEARCH INC., 100 TRADE CENTRE DRIVE, CHAMPAIGN, IL 61820, U.S.A.
E-mail address: `adams@wolfram.com`